

北京化工大学

深度学习课程设计

题 目 基于深度卷积网络的手写体数字识别

专 业 计算机科学与技术

姓 名 郑新威

目录

前　言

经过长时间的快速发展和转化，关于人工智能的研究产生了长足的发展。基于其研究而产生的成果被广泛的应用到社会的各个领域。在这些应用当中，一些领域如图像识别，人工智能的准确识别能力甚至超过了人类的识别能力，给人们的学习生活和整个社会的进步带来了很大的便利和改观。这其中反向传播算法的应用进一步推进了人工智能网络的大规模运用和快速发展。

第 1 章 绪论

1. 1 研究背景与意义

随着人工智能技术的发展进步和快速沉淀，近些年来，卷积神经网络(Convolutional Neural Network, CNN)广泛应用在加速器、解调器、图像分类、图像处理、语音识别等方面，尤其在图像分类方面表现出巨大的优势[1,2]。目前，卷积神经网络相关的算法种类繁多，而且卷积神经网络算法的特点就是运算量巨大。

1.2 研究现状

从一九六二年起，对卷积神经网络的研究就一直在进行着，Hubel 和 Wiesel 通过对实验动物猫的大脑中的视觉系统进行研究，总结出了感受野（Receptive Fields）的理论。一九八零年，日本的物理学家福岛邦彦首先给出了一种神经网络结构，这种神经网络包含卷积层和池化层，而在上世纪八十年代左右他又提出了一个 Attention 概念和网络。一九九八年，在日本物理学家福岛邦彦的研究基础上，Yann Lecun 提出了 LeNet-5 算法，把误差反向传播算法运用到神经网络结构的训练上，就形成了现今卷积神经网络最初的样子。

随着人工智能技术的不断发展，深度学习中的卷积神经网络逐渐成为领域的成熟算法之一。但与此同时，卷积神经网络的算法的计算复杂度也远高于传统的算法，网络的结构也越来越复杂。

第 2 章 卷积神经网络的处理

2.1 导言

由于人工神经网络具有高度非线性描述的特点，这个特点导致了他们被愈来愈广泛的研究和应用，在这些研究和应用当中主要的应用领域就是分类。分类实现的基础是特征分类，所以要进行分类就需要先提取样本的特征。

在常见的卷积神经网络中，通常是由输入层、卷积层、池化层、激活层、全连接层，按照一定的次序连接而构成。卷积神经网络的输入层实现的是整个神经网络的输入，在本设计中，训练和推理的数据为 30*30 像素的单通道灰度图。卷积神经网络的卷积层是深度学习中常用的一种神经网络层，其主要功能是对输入数据进行特征提取。在卷积层的内部包含多个卷积核，每个卷积核都对应一个权重系数和一个偏差量。这些卷积核可以有效地捕捉输入数据中的局部特征信息，并通过组合不同的卷积核来生成更高级别的特征表示。卷积神经网络中的池化层的池化操作是一种常用的操作，其主要功能是对输入矩阵进行降维处理。在池化操作中，通过对输入矩阵某一位置的相邻区域的总体统计特征进行平均或最大值的计算，得到该位置的输出。常见的池化操作主要有平均池化和最大池化。平均池化是将相邻区域的数值取平均值作为该位置的输出，可以有效地降低数据的维度，减少模型的复杂度和计算量。最大池化则是将相邻区域的最大值作为该位置的输出，可以保留数据的重要特征信息。简单来说，池化操作降低了特征图的节点个数，通过获取图像的主要特征来减少整个卷积神经网络的规模，减少网络的参数数量。池化层可以通过对输入矩阵进行降维处理，减少模型的复杂度和计算量，同时也可以保留数据的重要特征信息，提高模型的准确性和鲁棒性。由于卷积操作也是线性映射，要想给卷积神经网络加入非线性映射就需要加入激活函数，在本研究中，我们使用 Relu 函数来作为卷积神经网络的激活函数。在图像经过特征提取和信息过滤之后，我们还需要将提取到的信息特征输出，卷积神经网络的全连接层实现的功能就是将前面多次卷积得到的抽象化特征进行整合，最后样本归一化，对各个分类进行概率输出。

2.2 神经网络的训练和推理实现

上文中提到，本设计使用的训练和推理数据是 30*30 像素的单通道灰度图。由于

该样本的数据量较少，因此本设计采用了基于经典卷积神经网络 LeNet-5 提出了一种改进的网络模型 LeNet-3.3 来分析其在一体化实现方法中的表现。LeNet-3.3 网络采用除去池化层，增加网络复杂度与图像的特征信息量的方法，以此来研究复杂网络在此实现方法中的优势。LeNet-3.3 的网络模型如图 1-1 所示，该卷积神经网络具有六个网络层，前三层分别为卷积层，输入的数据是基于修改后的 MNIST 数据集单通道的 30*30 的像素特征图，每次经过卷积操作后取消后面的最大池化层，借助这种方法来保留所有的特征图信息；后面三层是全连接层，其中的最后一层完成图像的分类输出。这个卷积神经网络使用改进后的 ReLU 函数作为激活函数。最后通过 softmax 函数，对输出层的输出进行归一化处理。

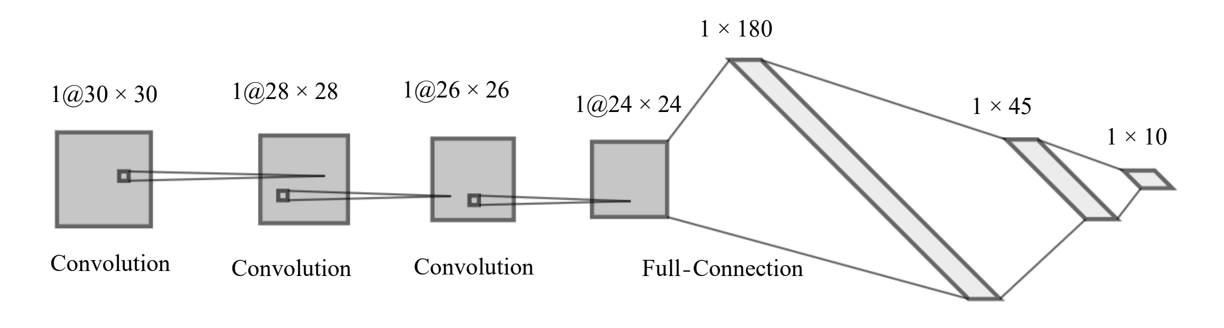


图 1-1　LeNet-3.3 网络模型

在整个卷积神经网络训练和推理的过程当中，操作系统可以对网络的学习率和优化器的策略等运算相关的参数进行动态的调整以提高模型的收敛速度，进而去优化网络的训练模型在所使用的推理数据集上的表现。操作系统也可以对参与预训练网络模型的权值文件进行读取操作，同时把相关的网络权值数据导入到可编程逻辑内部进行继续的训练，在网络全是数据进行训练的时候也可以得到生成的随机初试网络全集，把这些初始权值导入到可编程逻辑单元内即开始网络的训练。

使用 C 语言来重现构建神经网络中的前向传播模型和反向传播模型，这些模型包括卷积、池化、全连接以及激活函数等模块。前面提到，本设计采用的 LeNet-3.3 卷积神经网络的结构中含有六个网络层，这其中包含三个卷积层和三个全连接层，所以在进行神经网络的加速的时候需要针对这两个网络层进行优化。

2.2.1 卷积层结构设计

已知卷积层的前向传播可以表示为：

$$Z^{l}(\mathrm{x},\mathrm{y}) = \sum_{\mathrm{a}}\sum_{b} Z^{l-1}(x+a, y+b) * w^{l}(a,b) \qquad (2-1)$$

在此式中，$w^l(\mathrm{a,b})$表示的是第 1 层卷积的卷积核中所对应的坐标为(a,b)的元素，$z^l(\mathrm{x,y})$表示的是第 l 层卷积输出对应坐标为(x,y)的元素。式 2-1 是卷积的前向传播的关系式。卷积层前向传播的算法实现如下：

Algorithm 1: Convolution

w: 数据宽度

h: 数据高度

k: 卷积核维数

input_matrix: 卷积层输入数据

kernel: 卷积核参数

out_matrix: 卷积层输出数据

```
for(int i=0;i<w-k+1;i++)
    for(int j=0;j<h-k+1;j++){
        out_matrix[i*(h-k+1)+j]=0;
        for(int col=i;col<i+3;col++)
            for(int row=j;row<j+3;row++)
                out_matrix[i*(h-k+1)+j]+=input_matrix[col*h+row]*kernel[(col-i)*k+(row-j)];
    }
```

已知在第 l+1 层卷积的输出误差求第 I 层卷积的输出误差公式为：

$$\delta^l(\mathrm{x,y}) = \frac{\partial C}{\partial Z^l(\mathrm{x,y})} \qquad (2-2)$$

在这个公式中，误差$\delta^l(\mathrm{x,y})$指的是当前层未激活输出的$Z^l(x,y)$关于损失函数 C 的导数。在二维卷积才做中，每一层的 δ 误差都是一个二维矩阵。该矩阵的值代表的式在第 l 层坐标为(x,y)出的 δ 误差。根据函数求导的链式法则可以推到出来，式 2-2 可以表示为：

$$\delta^l(\mathrm{x,y}) = \sum_{x'}\sum_{y'} \frac{\partial C}{\partial Z^{l+1}(\mathrm{x',y'})} * \frac{\partial Z^{l+1}(\mathrm{x',y'})}{\partial Z^l(\mathrm{x,y})} \qquad (2-3)$$

已知式 2-1 的卷积前向传播公式和限制条件$\mathrm{x'+a=x}$ ， $\mathrm{y'+b=y}$ ，把已知条件带入到求导法则的公式当中，令$\mathrm{a'=-a}$，$\mathrm{b'=-b}$，即可得到：

$$\delta^l(\mathrm{x,y}) = \sum_{-a}\sum_{-b} \delta^{l+1}(\mathrm{x+a',y+b'}) * w^{l+1}(-a',-b') \qquad (2-4)$$

由以上公式可知，假定已知第 l+1 层的 δ 误差与第 l+1 层的卷积核，就可求出第

l 层的 δ 误差。并且把式 2-4 和卷积层的前向传播公式 2-1 相比可以推理出来：第 l 层的 δ 误差就是第 l+1 层的 δ 误差和第 l+1 层的卷积核旋转 180°之后的卷积结果。

但是由于需要考虑卷积操作中的输入矩阵和输出矩阵的大小也就是维度，假设卷积核的维度大小为 k，那么第 l 层的误差$\delta^l(x,y)$的维度大小就等同于卷积层输入矩阵Z^l的维度 i，第 l+1 层误差$\delta^{l+1}(x,y)$的维度就等同卷积层的输出矩阵 Z^{l+1} 的维度大小 j。根据卷积的前向传播公式可得 j= i-k+1 ，因此在网络反向传播时需要对第 l+1 层的误差 $\delta^{l+1}(x,y)$ 进行零值填充，在其各边填充 (k-1) 使得 $\delta^{l+1}(x,y)$维度为 j+2*(k-1)，在此基础上再进行卷积操作，之后输出的维度就是 j+2*(k-1)-k+1=j+k-1=i，符合第 l 层误差$\delta^l(x,y)$的矩阵维度 i 。

可以推导出，如果已知第 l+1 层卷积的输出误差那么求第 l 层卷积的输出误差就会需要分为三步进行实现：第一步是将卷积 w^l核旋转 180°；第二步就是将第 l+1 层的误差$\delta^{l+1}(x,y)$的各边填充 (k-1) 的零值；最后一步就是使用上述的卷积算法 Convolution 进行运算。旋转卷积核操作实现如下：

Algorithm 2: OverturnKernel

k: 卷积核维数

input_matrix: 旋转前的卷积核

out_matrix: 旋转后卷积核

```
1. for(int i=0;i<k;i++)
2.     for(int j=0;j<k;j++)
3.         output_matrix[(k-1-i)*k+(k-1-j)]=input_matrix[i*k+j];
```

矩阵零值填充的算法实现如下：

Algorithm 3: Padding

w: 待填充矩阵的宽度

_stride:填充步长

```
1. for(int i=0;i<w+2*_stride;i++)
2.     for(int j=0;j<w+2*_stride;j++){
3.         if((i>=stride)&&(j>=stride)&&(i<_stride+w)&&(j<_stride+w))
4.             output_matrix[i*(w+2*stride)+j]=input_matrix[(i-_stride)*w+(j-_stride)];
5.         else
6.             output_matrix[i*(w+2*_stride)+j]=0;   // 补 0
7.     }
```

综上，如果已知第 l 层的卷积输出结果的误差来去求该层的卷积核产生的误差也可以使用上述的卷积算法来实现。

2.2.2 全连接层结构设计

已知全连接的前向传播可以表示为：

$$Z^l = W^l * a^{l-1} \tag{2-5}$$

在式 2-5 中，W^l表示的是第 l 层的全连接权值矩阵，a^l表示的是第 l 层的全连接激活后的列向量输出，Z^l表示的是第 l 层全连接激活操作前的列向量输出结果。全连接矩阵的算法实现如下：

Algorithm 4: MatrixMultiPLy

h： 全连接输入向量维数

h_out：全连接输出向量维度

input_matrix:全连接输入矩阵

para_layer:全连接权重矩阵

out_matrix:全连接输出矩阵

```
for(int j=0;j<h_out;j++){
        output_matrix[j]=0;
        for(int i=0;i<h;i++)
                output_matrix[j]+=input_matrix[i]*para_layer[i*h_out+j];
}
```

已知第 l+1 层全连接输出误差是$\frac{\partial C}{\partial Z^{l+1}}$，可以得到第 l 层激活后的输出误差为$\frac{\partial C}{\partial a^l}$：

$$\frac{\partial C}{\partial a^l} = \frac{\partial C}{\partial Z^{l+1}} * \frac{\partial Z^{l+1}}{\partial a^l} \tag{2-6}$$

将全连接的前向传播式 2-5 带入并且由矩阵的求导法则可知:

$$\frac{\partial C}{\partial a^l} = (W^{l+1})^T * \frac{\partial C}{\partial Z^{l+1}} \tag{2-7}$$

计算矩阵的梯度的算法实现如下：

续表

Algorithm 5: CalculateMatrixGrad

w： 全连接权重矩阵宽度

h： 全连接权重矩阵高度

input_matrix:全连接权重矩阵

grad：全连接输出梯度

out_matrix:全连接输入梯度

```
for(int i=0;i<w;i++){
    output_matrix[i]=0;//清空梯度便于累加操作
    for(int j=0;j<h;j++)
        output_matrix[i]+=input_matrix[i*h+j]*grad[j];
}
```

已知第1层的全连接输出误差是$\frac{\partial C}{\partial z^l}$，那么第1层的全连接权值矩阵的输出误差 $\frac{\partial C}{\partial z^l}$ 为：

$$\frac{\partial C}{\partial W^l}=\frac{\partial C}{\partial Z^l}*\frac{\partial Z^l}{\partial w^l} \qquad (2-8)$$

将全连接的前向传播公式（式2-5）带入并且由矩阵的求导规则可知：

$$\frac{\partial C}{\partial W^l}=\frac{\partial C}{\partial Z^l}*(a^{l-1})^T \qquad (2-9)$$

求全连接权值矩阵的输出误差的算法实现如下：

Algorithm 6: MatrixBackPropagationMultiPLy

w： 全连接权重矩阵宽度

h： 全连接权重矩阵高度

input_matrix:全连接输入梯度

grad：全连接输出梯度

r_grad：全连接权重矩阵梯度

```
for(int i=0;i<w;i++)
    for(int j=0;j<h;j++)
        r_grad[i*h+j]=input_matrix[i]*grad[j];
```

对于全连接层来说，这一层的输入和输出的特征图和梯度都是以为向量，输入和出处的特征图的权值和梯度都是二维向量。在实现全连接网络模块的前向传播时，把特征图和权值进行加权运算之后，再通过激活函数进行激活就可以得到了全连接模块的输出结果。在实现神经网络的全连接模型的反向传播的时候，输入的梯度经过激活函数的导数之后与权值矩阵的转置进行加权运算后就可以得到反向传播输出的梯度。

2.2.3 池化层结构设计

神经网络的池化层位于连续的卷积层之间，池化层起到的主要作用是在神经网络进行运算的时候保留住输入数据的主要特征同时尽可能的减少网络中的参数，进而减

少运算过程中的运算量，防止网络出现过拟合的现象同时进一步提高网络模型的泛化能力。其中常见的池化操作包括平均池化和平均池化。平均池化指的是将相邻区域的数值取平均值来作为该位置的输出。同理，最大池化指的是选取输入矩阵的最大值来作为该位置的输出，这两种不同的池化策略起到的作用也不同，其中最大池化有助于保留图像的所有特征（包含边缘特征），而平均池化有利于保留图像的背景特征。基于两种不同的池化策略产生的作用，我们需要对数据集的图像特征进行判断进而实现手写体数字的识别功能，因此我们使用最大池化的池化策略。

数据在前向传播的时候，使用池化核来扫描数据并且选取该区域中的最大值作为该区域的输出结果，同时记录下该最大值在原数据中的位置以便实现后面的反向传播操作。其中，池化层的前向传播的算法实现如下：

Algorithm 7: MaxPool2d

w： 池化前数据宽度

h： 池化前数据高度

k： 池内核维度

input_matrix:池化前数据

output_tmatrix:池化后数据

locate_matrix：池化中位置矩阵

```
for(int i=0;i<w/k;i++)
        for(int j=0;j<h/k;j++){
            int max_num=-999;
            for(int col=i*k;col<(i+1)*k;col++)
                for(int row=j*k;row<(j+1)*k;row++)
                    if(input_matrix[col*h+row]>max_num){
                        max_num=input_matrix[col*h+row];
                        locate_matrix[i*(h/k)+j]=col*h+row;
                    }
             output_matrix[i*(h/k)+j]=max_num;
        }
```

上文提到，最大池化的策略就是使用扫描区域的最大值来作为该区域的输出结果，因此数据的输出的梯度就是该最大元素的梯度，其余元素的梯度的值为零。在损失梯度反向传播的时候，找到对应的位置赋值，其余的梯度的值为零。因此，池化层的梯度反向传播的算法实现如下：

Algorithm 8: MaxPooBackPropagation

w： 池化前数据宽度

h： 池化前数据高度

k： 池内核维度

input_matrix:池化后梯度

outputmatrix：池化前梯度

locate_matrix：池化中位置矩阵

```
for(int col=0;col<w;col++)
        for(int low=0;low<h;low++)
                output_matrix[col*h+low]=0;
int current_locate;
for(int i=0;i<w/k;i++)
        for(int j=0;j<h/k;j++){
                current_locate = locate_matrix[i*(h/k)+j];
                output_matrix[current_locate]=input_matrix[i*(h/k)+j];
        }
```

2.3 小结

在本章中，主要介绍了卷积神经网的特点，其特点就是具有高度的非线性描述的特点，基于该神经网络的非线性描述的特点，目前有越来越多关于他们的研究和分类。接着介绍了卷积神经网络的常见模型，在常见的卷积神经网络中，通常是由输入层、卷积层、池化层、激活层、全连接层，按照一定的次序连接而构成。接着简单介绍了卷积神经网络的各层实现的功能，最后完成样本概率的分类输出。

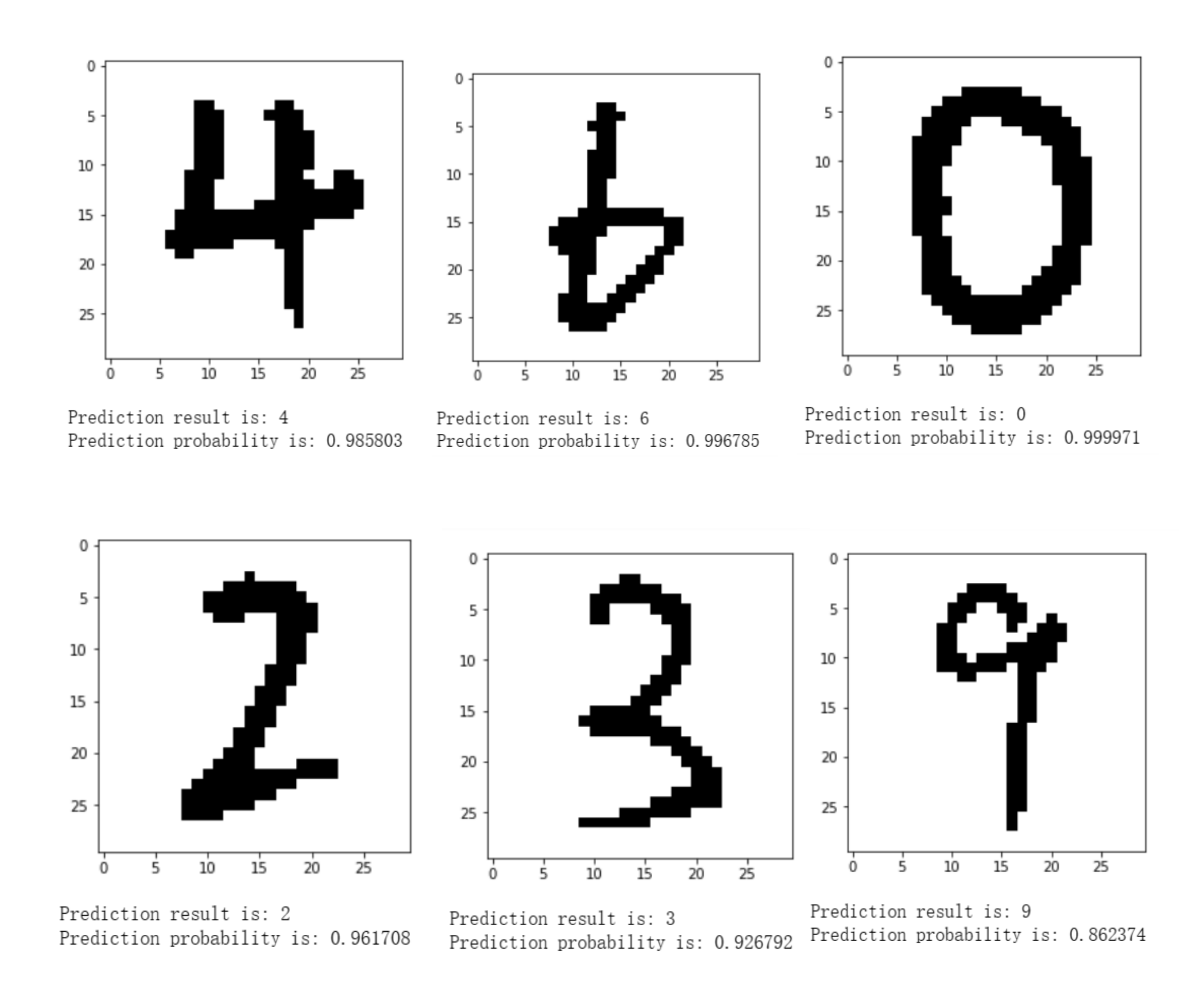


图 2-1 数据集上的部分识别结果

第 3 章 结论

首先，本设计仔细探究了卷积神经网络在训练和推理过程中的数据流结构。实验结果表明，对 MNIST 手写体数字验证集的识别准确率为 93.637%，单张图片的训练时长为 23.2ms，单张图片的推理时长为 4.51ms 通过使用片上多处理器系统得可编程逻辑来实现卷积神经网络的训练和推理极大的降低了神经网络训练和推理过程中的消耗并且显著提高了相关设备的资源利用率，最大程度上的减少了处理器的运算负载，平衡了算力和灵活性。

卷积神经网络在人工智能领域取得了巨大的成就，为了适应功耗和成本敏感领域的相关应用，基对于该领域的研究具有非常可观的研究价值，但是本设计依然很多的可以改进的空间：

算法实现方面，本设计所使用的卷积神经网络模型较小，随着应用场景越来越复杂和使用准确性的提高，将来的网络模型还可以朝着更大的网络模型发展。

参考文献

[1] Y. LeCun,L. Bottou,Y. Bengio,P. Haffner. Gradient-based learning apPLied to document recognition[J] . Proceedings of the IEEE . 1998 (11)

[2] Guo K, Sui L, Qiu J, et al. Angel-eye: A comPLete design flow for mapping CNN onto embedded FPGA[J]. IEEE transactions on computer-aided design of integrated circuits and systems, 2017, 37(1): 35-47

[3] Zheng Y, He B, Li T. Research on the Lightweight DePLoyment Method of Integration of Training and Inference in Artificial Intelligence[J]. ApPLied Sciences, 2022, 12(13): 6616.